Amendments to the Claims:

This listing of claims will replace all prior version, and listings of claims in the application:

Listing of Claims:

1. (Currently Amended) A method of generating a random permutation of a block of data elements, comprising the steps of:

(a) generating an input block of 2^N data elements, wherein each data element occupies a particular position in the input block, each data element being represented by plural symbols; and

(b) for each data element in the input block, conditionally changing the value of individual symbols of the data element in accordance with random data to form an output data element in a corresponding position in an output block of data elements, the output data element being one of the data elements in the input block, wherein each data element is mapped from a position in the input block to a position in the output block, such that the output block of data elements is a random permutation of the input block of data elements and the input block of data elements is permuted using no more than $N \cdot 2^{N-1}$ random bits.

2. (Original) The method of claim 1, wherein, for each symbol of a data element, step (b) includes retrieving a random symbol and determining a value of a corresponding symbol of the output data element as a function of the symbol of the data element and the random symbol.

3. (Original) The method of claim 2, wherein the corresponding symbol of the output data element is the exclusive-OR of the symbol of the data element and the random symbol.

4. (Previously Presented) The method of claim 2, wherein (b) includes retrieving the random symbol using an address comprising symbols from at least one of the data element and the output data element.

5. (Original) The method of claim 4, wherein:

each data element in the input block comprises N symbols T_0, T_1, ..., T_{N-1}, where N is an integer greater than one, and each output data element in the output data block comprises N symbols H_0, H_1, ..., H_{N-1};

for each data element T_i, for i = 0 to N-1, step (b) includes retrieving a random symbol R_i using an (N-1)-symbol memory address comprising symbols of the data element whose indices are greater than i and symbols of the output data element whose indices are less than i; and

determining symbol H_i as the exclusive-OR of T_i and R_i, for i = 0 to N-1.

6. (Original) The method of claim 4, wherein:

each data element in the input block comprises N symbols T_0, T_1, ..., T_{N-1}, where N is an integer greater than one, and each output data element in the output data block comprises N symbols H_0, H_1, ..., H_{N-1};

for each data element T_i, for i = 0 to N-1, step (b) includes retrieving a random symbol R_i using a memory address H_{i-1}, ..., H_{i-N+1}, T_{i+N-1}, ..., T_{i+1} where indices for the output data block symbols H are non-negative and indices for the data element symbols T are less than N; and

determining symbol H_i as the exclusive-OR of T_i and R_i, for i = 0 to N-1.

7. (Original) The method of claim 1, further comprising:

(c) altering the random data; and

(d) repeating steps (a) and (b) with the altered random data to produce an output block that is a different random permutation of the input block of data elements.

8. (Original) The method of claim 7, wherein the random data is altered after every repetition of steps (a) and (b), such that an order of data element in every output block is independent of the order of data elements in other output blocks, thereby forming an output sequence of data elements comprising varying permutations of the input block of data elements.

9. (Currently Amended) ~~The method of claim 7,~~ A method of generating a random permutation of a block of data elements, comprising:

(a) generating an input block of data elements, wherein each data element occupies a particular position in the input block, each data element being represented by plural symbols;

(b) for each data element in the input block, conditionally changing the value of individual symbols of the data element in accordance with random data to form an output data element in a corresponding position in an output block of data elements, the output data element being one of the data elements in the input block, wherein each data element is mapped from a position in the input block to a position in the output block, such that the output block of data elements is a random permutation of the input block of data elements;

(c) altering the random data; and

(d) repeating (a) and (b) with the altered random data to produce an output block that is a different random permutation of the input block of data elements, wherein only a subset of the random data is altered, such that ordering of data elements in consecutive output blocks is not independent.

10. (Currently Amended) The method of claim 9 ~~7~~, wherein the random data is altered less frequently than every repetition of steps (a) and (b), such that a same permutation is produced in plural consecutive output blocks.

11. (Original) The method of claim 1, wherein said symbols are binary bits.

12. (Original) The method of claim 1, wherein the data elements are numbers and the input block is a sequence of incrementing numbers.

13. (Original) The method of claim 1, wherein the data elements correspond to transmission frequencies in a frequency-hopping communication system, and the output sequence is a hop code sequence.

14. (Original) The method of claim 1, wherein each data element occurs once in the input block.

15. (Currently Amended) ~~The method of claim 1, further comprising the step of:~~

A method of generating a random permutation of a block of data elements, comprising:

(a) generating an input block of data elements, wherein each data element occupies a particular position in the input block, each data element being represented by plural symbols;

(b) for each data element in the input block, conditionally changing the value of individual symbols of the data element in accordance with random data to form an output data element in a corresponding position in an output block of data elements, the output data element being one of the data elements in the input block, wherein each data element is mapped from a position in the input block to a position in the output block, such that the output block of data elements is a random permutation of the input block of data elements; and

(c) determining whether the output data element formed in (b) has a predetermined value and, if so, substituting a data element having a different value for the output data element, such that the output block does not include the output data element having the predetermined value.

16. (Canceled)

17. (Original) The method of claim 1, further comprising the step of:

(c) forming a truncated permutation that is a permutation of only a subset of the data elements in the input block.

18. (Currently Amended) ~~The method of claim 17,~~ A method of generating a random permutation of a block of data elements, comprising:

(a) generating an input block of data elements, wherein each data element occupies a particular position in the input block, each data element being represented by plural symbols;

(b) for each data element in the input block, conditionally changing the value of individual symbols of the data element in accordance with random data to form an output data element in a corresponding position in an output block of data elements, the output data element being one of the data elements in the input block, wherein each data element is mapped from a position in the input block to a position in the output block, such that the output block of data elements is a random permutation of the input block of data elements; and

(c) forming a truncated permutation that is a permutation of only a subset of the data elements in the input block, wherein the truncated permutation includes data elements occupying selected positions in the output block, which originated from selected positions in the input block, and excludes data elements occupying excluded positions in the output block, which originated from excluded positions in the input block, wherein step (c) includes:

(c1) after step (b), identifying all selected output positions occupied by a data element from an excluded input position;

(c2) after step (b), identifying all excluded output positions occupied by a data element from a selected input position; and

(c3) remapping the data elements occupying the excluded output positions identified in step (c2) into the selected output positions identified in step (c1).

19. (Currently Amended) An apparatus for generating a random permutation of a block of data elements, comprising:

an input device configured to supply an input block of 2^N data elements, wherein each data element occupies a particular position in the input block, each data element being represented by plural symbols;

a random code memory device for storing random data; and

a permutation logic unit responsive to the data elements from said input register to generate output data elements, wherein, for each data element in the input block, said permutation logic unit conditionally changes the value of individual symbols of the data element in accordance with random data retrieved from said random code memory to form an output data element in a corresponding position in an output block of data elements, the output data element being one of the data elements in the input block, wherein each data element is mapped from a position in the input block to a position in the output block, such that the output block of data elements is a random permutation of the input block of data elements and the permutation logic unit permutes the input block of data elements using no more than $N \cdot 2^{N-1}$ random bits.

20. (Original) The apparatus of claim 19, further comprising an output register which receives each output data element generated by said permutation logic unit.

21. (Original) The apparatus of claim 19, wherein said input device is a counter that periodically supplies an incremented number in response to a periodic signal.

22. (Original) The apparatus of claim 19, wherein said random code memory device comprises a plurality of random code memories each storing a plurality of random data bits.

23. (Original) The apparatus of claim 19, wherein, for each symbol of a data element, said permutation logic unit retrieves a random symbol from said random code memory device and determines a value of a corresponding symbol of the output data element as a function of the symbol of the data element and the random symbol.

24. (Original) The apparatus of claim 23, wherein said permutation logic unit determines the corresponding symbol of the output data element is the exclusive-OR of the symbol of the data element and the random symbol.

25. (Original) The apparatus of claim 23, wherein said permutation logic unit retrieves the random symbol from said random code memory device using an address comprising symbols from at least one of the data element and the output data element.

26. (Original) The apparatus of claim 25, wherein:

each data element in the input block comprises N symbols T_0, T_1, ..., T_{N-1}, where N is an integer greater than one, and each output data element in the output data block comprises N symbols H_0, H_1, ..., H_{N-1}; and

for each data element T_i, for i = 0 to N-1, said permutation logic unit retrieves a random symbol R_i from said random code memory device using an (N-1)-symbol memory address comprising symbols of the data element whose indices are greater than i and symbols of the output data element whose indices are less than i; and determines symbol H_i as the exclusive-OR of T_i and R_i, for i = 0 to N-1.

27. (Original) The apparatus of claim 26, wherein said random code memory device comprises N random code memories respectively corresponding to the N symbols of each data element, wherein random code memory i supplies random symbol R_i, for i = 0 to N-1.

28. (Original) The apparatus of claim 25, wherein:

each data element in the input block comprises N symbols T_0, T_1, ..., T_{N-1}, where N is an integer greater than one, and each output data element in the output data block comprises N symbols H_0, H_1, ..., H_{N-1}; and

for each data element T_i, for i = 0 to N-1, said permutation logic unit retrieves a random symbol R_i from said random code memory device using a memory address H_{i-1}, ..., H_{i-N+1}, T_{i+N-1}, ..., T_{i+1} where indices for the output data block symbols H are non-negative and indices for the data element symbols T are less than N; and determines symbol H_i as the exclusive-OR of T_i and R_i, for i = 0 to N-1.

29. (Original) The apparatus of claim 19, wherein

said input device repeatedly supplies the input block of data elements to said permutation logic unit;

said random code memory device periodically stores altered random data; and

said permutation logic unit repeatedly forms output blocks of data elements from the input blocks of data elements using the altered random data, such that the output blocks of data elements represent different random permutations of the input block of data elements.

30. (Original) The apparatus of claim 29, wherein the random data is altered after every repetition of the input block of data elements, such that an order of data elements in every output block is independent of the order of data elements in other output blocks, thereby forming an output sequence of data elements comprising varying permutations of the input block of data elements.

31. (Currently Amended) ~~The apparatus of claim 29,~~

An apparatus for generating a random permutation of a block of data elements, comprising:

an input device configured to supply an input block of data elements, wherein each data element occupies a particular position in the input block, each data element being represented by plural symbols;

a random code memory device for storing random data; and

a permutation logic unit responsive to the data elements from said input register to generate output data elements, wherein, for each data element in the input block, said permutation logic unit conditionally changes the value of individual symbols of the data element in accordance with random data retrieved from said random code memory to form an output data element in a corresponding position in an output block of data elements, the output data element being one of the data elements in the input block, wherein each data element is mapped from a

position in the input block to a position in the output block, such that the output block of data elements is a random permutation of the input block of data elements, wherein:

the input device repeatedly supplies the input block of data elements to said permutation logic unit;

the random code memory device periodically stores altered random data;

the permutation logic unit repeatedly forms output blocks of data elements from the input blocks of data elements using the altered random data, such that the output blocks of data elements represent different random permutations of the input block of data elements; and

only a subset of the random data is altered, such that ordering of data elements in consecutive output blocks is not independent.

32. (Currently Amended) The apparatus of claim 31 ~~29~~, wherein the random data is altered less frequently than every repetition of the input block of data elements, such that a same permutation is produced in plural consecutive output blocks.

33. (Original) The apparatus of claim 19, wherein said symbols are binary bits.

34. (Original) The apparatus of claim 19, wherein the data elements are numbers and the input block is a sequence of incrementing numbers.

35. (Original) The apparatus of claim 19, wherein the data elements correspond to transmission frequencies in a frequency-hopping communication system, and the output sequence is a hop code sequence.

36. (Original) The apparatus of claim 19, wherein each data element occurs once in the input block.

37. (Currently Amended) ~~The apparatus of claim 19, further comprising:~~

An apparatus for generating a random permutation of a block of data elements, comprising:

an input device configured to supply an input block of data elements, wherein each data element occupies a particular position in the input block, each data element being represented by plural symbols;

a random code memory device for storing random data;

a permutation logic unit responsive to the data elements from said input register to generate output data elements, wherein, for each data element in the input block, said permutation logic unit conditionally changes the value of individual symbols of the data element in accordance with random data retrieved from said random code memory to form an output data element in a corresponding position in an output block of data elements, the output data element being one of the data elements in the input block, wherein each data element is mapped from a position in the input block to a position in the output block, such that the output block of data elements is a random permutation of the input block of data elements;

means for determining whether the output data element formed by said permutation logic unit has a predetermined value; and

means for substituting a data element having a different value for the output data element if the output data element has the predetermined value, such that the output block does not include the output data element having the predetermined value.

38. (Canceled)

39. (Original) The apparatus of claim 19, wherein said permutation logic unit forms a truncated permutation that is a permutation of only a subset of the data elements in the input block.

40. (Original) The apparatus of claim 39, wherein:

the truncated permutation includes data elements occupying selected positions in the output block, which originated from selected positions in the input block, and excludes data elements occupying excluded positions in the output block, which originated from excluded positions in the input block; and

after initially mapping all of the data element in the input block into the output block, said permutation logic unit identifies all selected output positions occupied by a data element from an excluded input position; identifies all excluded output positions occupied by a data element from a selected input position; and re-maps the data elements occupying the identified excluded output positions into the identified selected output positions.

41-44. (Canceled)